

Rapport Jeu Moulin

Réalisé par :

AJALE Saad
BENHADDOU Lahcen
ERRAZI Fatima-Zahra

Professeure encadrante:

Madame ADDOU Malika

Élément de module:

Résolution de Problèmes

Classe:

1^{ère} année Génie Informatique

Plan

Introduction.....
Modélisation.....
Résolution du problème par MiniMax.....
Résolution du problème par Alpha-Beta.....
Application console.....
Comparaison des deux algorithmes.....
Base de données.....
Application graphique.....
Conclusion.....

Introduction

Le jeu du moulin, ou "merels", est un jeu de société ancien dont les origines remontent à environ 1400 av. J.-C. en Égypte antique. Populaire dans la Grèce antique et l'Empire romain, où il était connu sous le nom de "mola" ou "milliarium", il s'est répandu en Europe au Moyen Âge, devenant un divertissement apprécié aussi bien par les nobles que par les paysans.

Au fil des siècles, le jeu du moulin a conservé son attrait grâce à sa simplicité et à sa profondeur stratégique. Aujourd'hui, nous souhaitons revitaliser ce jeu classique en créant une application moderne. Ce projet vise à rendre le jeu du moulin accessible à une nouvelle génération de joueurs à travers le monde, honorant son riche héritage tout en tirant parti des technologies contemporaines.

Dans ce sens, nous optons pour l'utilisation des algorithmes *minimax* et *alpha-bêta* pour la réalisation de la partie intelligente permettant le jeu contre la machine.

Ainsi, une interface graphique est importante pour donner plus de vivacité au jeu. En plus, nous avons conçu une base de données pour le jeu.

Le code de ce projet est comme suit :

- ✧ Les codes Minimax et Alpha-Beta sont implémentés à l'aide du *langage C*.
- ✧ L'interface graphique est implémentée à l'aide du *langage C++*.

Modélisation

➤ Espace d'états

Le plateau du jeu est sous forme d'une matrice de type caractère de taille 7*7 .

Donc avec 49 positions :

24 positions, qui peuvent contenir les pions, initialisées à vide tel que:

- ✓ Si la position contient pion noir correspond à '1'
- ✓ Si la position contient pion blanc correspond à '2'
- ✓ Si la position est vide correspond à 'V'

Le reste des positions ne sont pas modifiées, contenant 'N'.

Les caractères ci-dessus sont des définies comme des macros *#define* :

JETON1 -> '1'

JETON2 -> '2'

VIDE -> 'V'

NONE -> 'N'

Le nombre de pions noirs perdus sont modélisés par une variable nommée `jetonGagne2`.

Le nombre de pions blancs perdus sont modélisés par une variable nommée `jetonGagne1`.

➤ État initial

L'état initial correspond à un plateau vide et une une matrice toute initialisée à 'V'

➤ État final

L'état final correspond à une matrice ne contenant que :

✓ deux pions noirs + $\text{jetonGagne2} = 7$

(dans ce cas noir est noté perdant)

ou

✓ deux pions blanc + $\text{jetonGagne1} = 7$

(dans ce cas blanc est noté perdant) .

➤ Règles du jeu

Le jeu commence avec un plateau vide entre deux joueurs (Max et Min dans le cas du jeu contre la machine).

- Les joueurs doivent placer à tour de rôle leurs pions (neuf pions au maximum) sur un point d'intersection libre du plateau, suivant la couleur qui leur est attribuée noire ● ou blanche ○.
- Le but est de faire un « **moulin : trois pions alignés** ». Si un joueur forme un moulin, il retire un pion à son adversaire (en dehors d'un moulin éventuel).
- Quand les pions sont tous posés, les joueurs peuvent les glisser d'un point d'intersection à un autre point de la ligne (un mouvement à la fois).
- Quand un joueur n'a que trois pions, il peut sauter d'un point à un autre.
- Les pions formant un moulin sont intouchables par l'adversaire. Mais s'ils sont déplacés par le propriétaire, ils deviennent vulnérables.
- Le jeu continue ainsi jusqu'à ce qu'un joueur n'a que deux pions en sa possession. Dans ce cas, son adversaire est déclaré gagnant.

➤ Fonction heuristique h

Cette fonction est utilisée dans le cas du jeu contre la machine. Elle estime le meilleur coup à choisir par Min pour minimiser les possibilités de gain pour Max.

La fonction utilisée dans notre projet est:

$$(\text{Nombre de moulins formés par Max} - \text{Nombre de moulins formés par Min}) + 2 * (\text{nombre de Jetons perdus par Max} - \text{nombre de Jetons perdus par Min})$$

Le coefficient 2 est justifié par le fait qu'on donne plus de poids aux pions consommés qu'au nombre de moulins formés.

L'algorithme de Minimax

En suivant les règles spécifiques au jeu du Moulin, l'algorithme de Minimax est utilisé pour déterminer les meilleurs coups pour un joueur en évaluant systématiquement les coups possibles et les réponses optimales de l'adversaire.

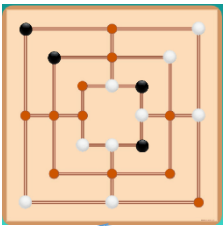
Le joueur Max (qui veut maximiser son avantage) commence son tour.

Une profondeur maximale est définie pour limiter la recherche (profondeur $p = 4$ dans notre cas)

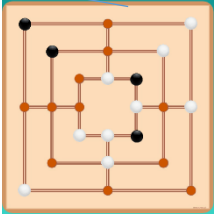
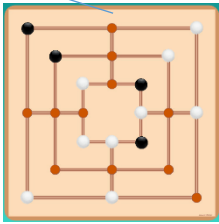
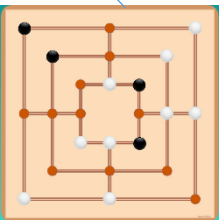
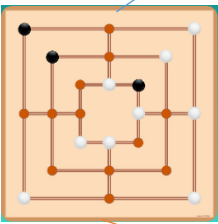
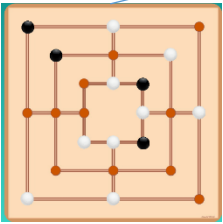
- Pour le joueur Max : Calculer la valeur de chaque coup possible (placement ou mouvement) et choisir le coup avec la valeur maximale.
- Pour le joueur Min : Calculer la valeur de chaque coup possible et choisir le coup avec la valeur minimale.

La valeur retournée par la fonction Minimax pour le **joueur Max sera la valeur maximale** des coups possibles.

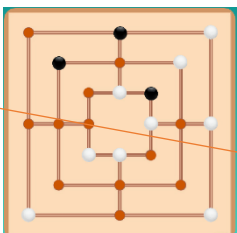
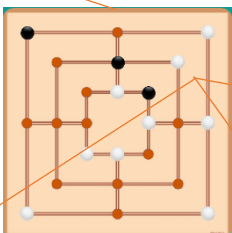
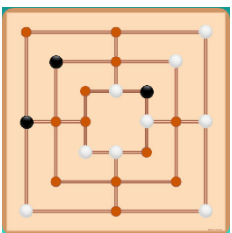
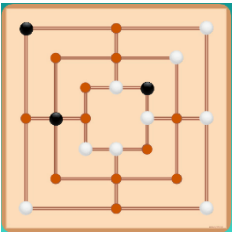
Pour le **joueur Min, ce sera la valeur minimale.**



Max

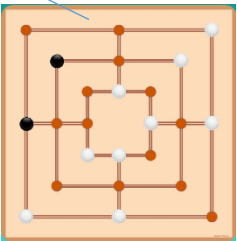


Min



Max

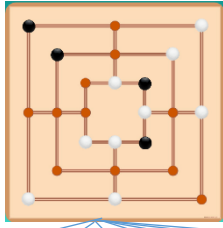
Etat solution pour max



L'algorithme de Alpha-Beta

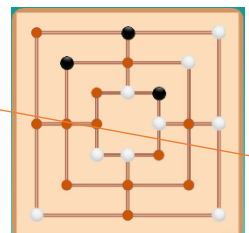
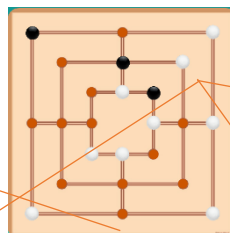
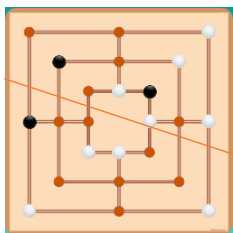
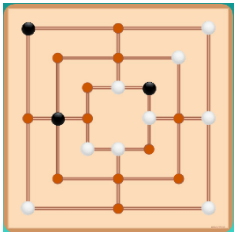
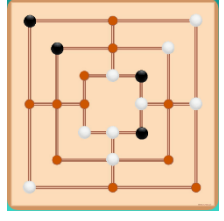
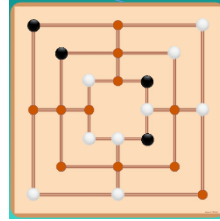
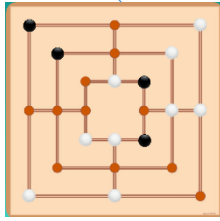
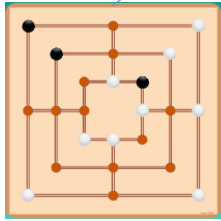
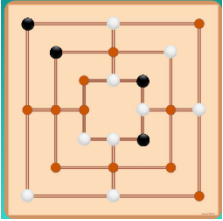
L'algorithme Alpha-Beta est une optimisation de l'algorithme Minimax qui permet de réduire le nombre de nœuds à explorer dans l'arbre de jeu, rendant la recherche plus efficace. Cette technique utilise deux valeurs, alpha et beta, pour élaguer les branches de l'arbre de jeu qui ne peuvent pas influencer la décision finale.

- ✓ Alpha est initialisé à $-\infty$ et Beta à $+\infty$.
- ✓ Une profondeur maximale est définie pour limiter la recherche (profondeur $p = 5$ dans notre cas).



Max

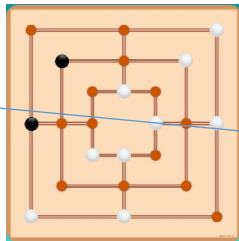
Min



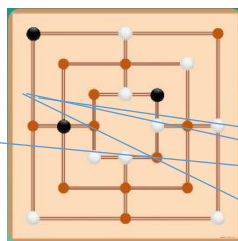
Max

Etat solution pour max

H=14



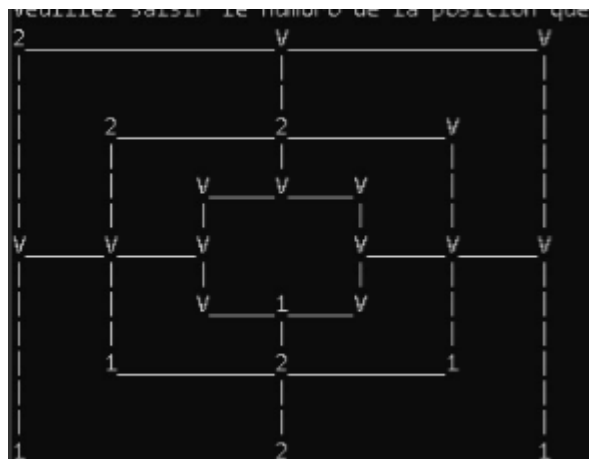
H=12



Application console

Nous avons réalisé une application console du jeu Moulin.

Le plateau du jeu est affiché sous la forme suivante:



Comparaison des deux algorithmes

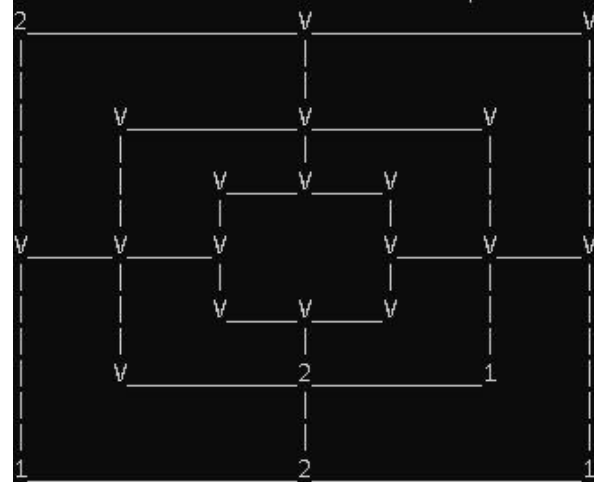
A l'implémentation de l'application console, nous avons calculé pour chacun des deux algorithmes le nombre de nœuds parcourus jusqu'à l'état final.

Le résultat pour Minimax & Alpha-Beta:

Pour ue profondeur de $p = 4$

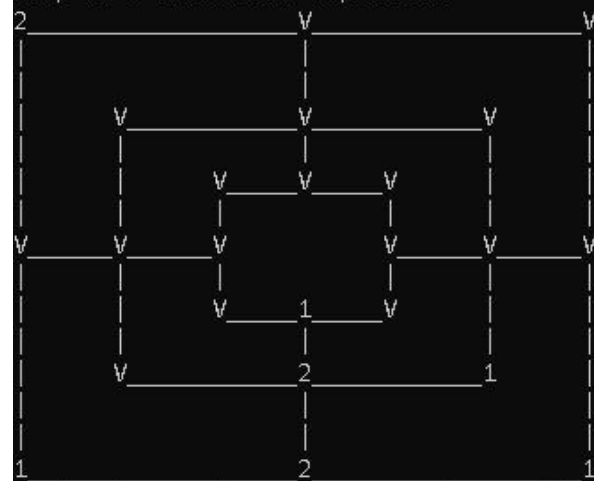
19. (5, 3)

Veillez saisir le numero de la position que vous voulez srlectionner : 19



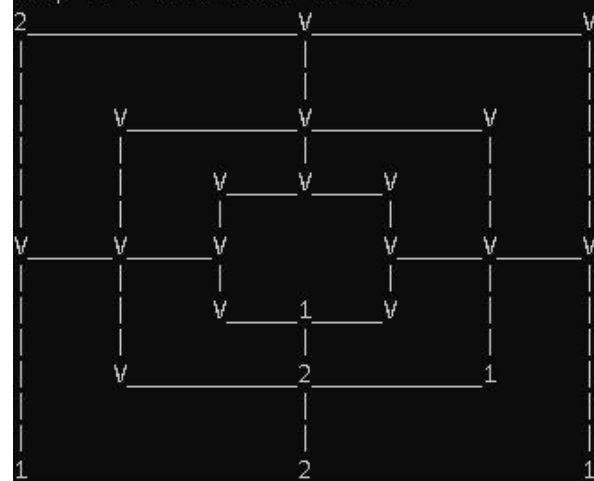
Nombre de noeuds generes par alphabeta : 84078

Coup de l'ordinateur Alphabeta:



Nombre de noeuds generes par minmax: 115252

Coup de l'ordinateur minmax:



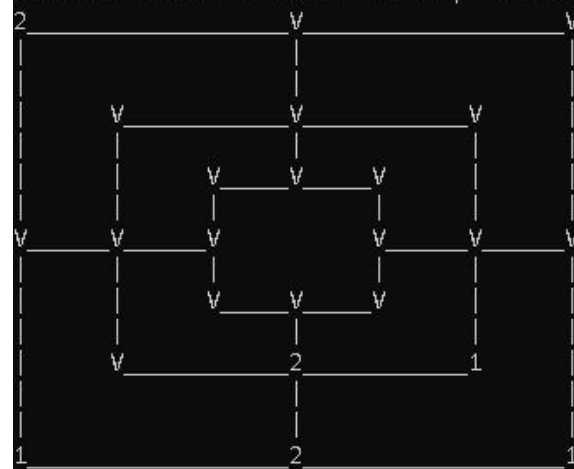
Pour une profondeur $p = 5$

17. (3, 3)

```

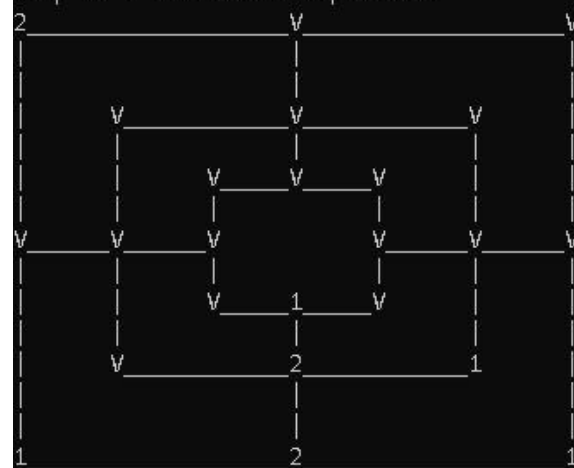
Veillez saisir le numero de la position que vous voulez srlectionner : 19

```



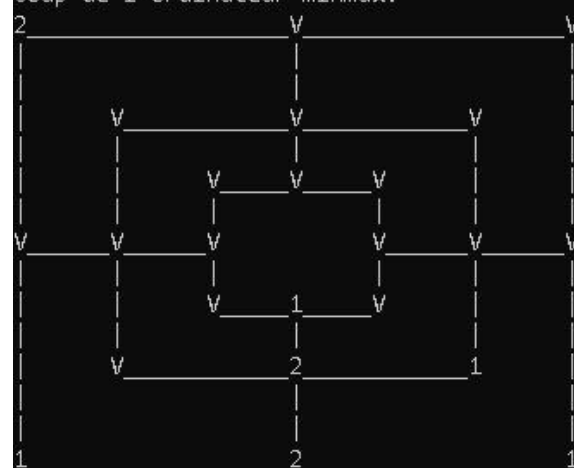
Nombre de noeuds generes par alphabeta : 635473

Coup de l'ordinateur Alphabet:



Nombre de noeuds generes par minmax: 1836040

Coup de l'ordinateur minmax:











Indiquez où vous souhaitez insérer :

=> Nous remarquons que le nombre de nœuds de dans Minimax est beaucoup plus inférieur à celui de Alpha-beta.

Base de données

Nous avons conçu pour les joueurs dans le cas de jeu en ligne, c'est-à-dire avec deux joueurs qui jouent à distance une base de données où leurs informations sont stockées.

La table suivante stocke le nom, prénom, nom d'utilisateur, mot de passe, nombre de fois qu'il a joué, nombre de fois qu'il a gagné et son état s'il est actif ou non.

#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra	Action
<input type="checkbox"/>	1 nom	varchar(50)	utf32_bin		Oui	NULL			 Modifier  Supprimer  Plus
<input type="checkbox"/>	2 prenom	varchar(50)	utf32_bin		Oui	NULL			 Modifier  Supprimer  Plus
<input type="checkbox"/>	3 username	 varchar(50)	utf32_bin		Non	Aucun(e)			 Modifier  Supprimer  Plus
<input type="checkbox"/>	4 password	varchar(255)	utf32_bin		Oui	NULL			 Modifier  Supprimer  Plus
<input type="checkbox"/>	5 partieJoue	int(11)			Oui	NULL			 Modifier  Supprimer  Plus
<input type="checkbox"/>	6 partieGagne	int(11)			Oui	NULL			 Modifier  Supprimer  Plus
<input type="checkbox"/>	7 etat	varchar(50)	utf32_bin		Oui	NULL			 Modifier  Supprimer  Plus

La table suivante stocke l'état et l'avancement du jeu dans une session:

#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra	Action
<input type="checkbox"/>	1 matrice	varchar(50)	utf32_bin		Non	Aucun(e)			 Modifier  Supprimer  Plus
<input type="checkbox"/>	2 jetonGagne1	int(11)			Non	Aucun(e)			 Modifier  Supprimer  Plus
<input type="checkbox"/>	3 jetonGagne2	int(11)			Non	Aucun(e)			 Modifier  Supprimer  Plus
<input type="checkbox"/>	4 tour	int(11)			Non	Aucun(e)			 Modifier  Supprimer  Plus
<input type="checkbox"/>	5 utilisateur1	 varchar(100)	utf32_bin		Non	Aucun(e)			 Modifier  Supprimer  Plus
<input type="checkbox"/>	6 utilisateur2	 varchar(100)	utf32_bin		Non	Aucun(e)			 Modifier  Supprimer  Plus

Application graphique

L'application graphique est implémentée en utilisant la bibliothèque *sfml* en C++.

Ce qui a nécessité l'utilisation de plusieurs notions :

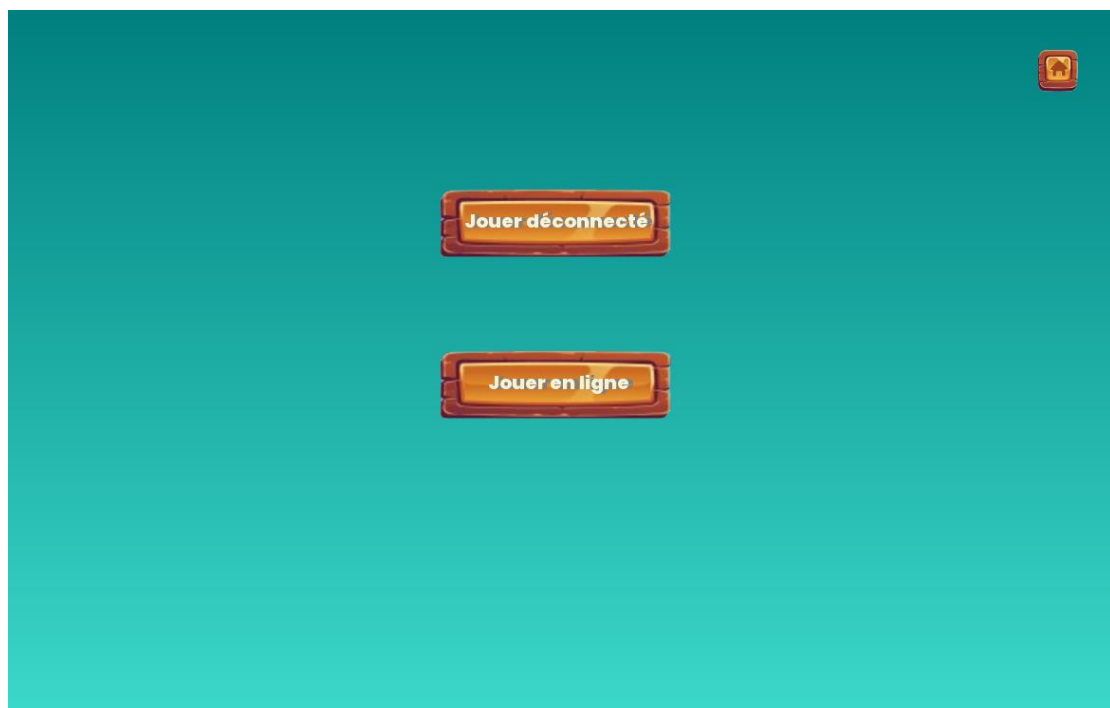
- ◆ Les fenetres et leurs paramètres
- ◆ La manipulation des sprites
- ◆ La manipulation des formes
- ◆ La manipulation du texte
- ◆ La détection des mouvements de la souris
- ◆ La détection des touches du clavier
- ◆ Le défilement

L'application console contient plusieurs fenetres, selon le type de jeu choisi par le joueur.

Le menu:



Choix du type de jeu:



Interface de connexion:

Formulaire de Connexion




Username:

Mot de passe:

Interface d'inscription:

Formulaire d'inscription



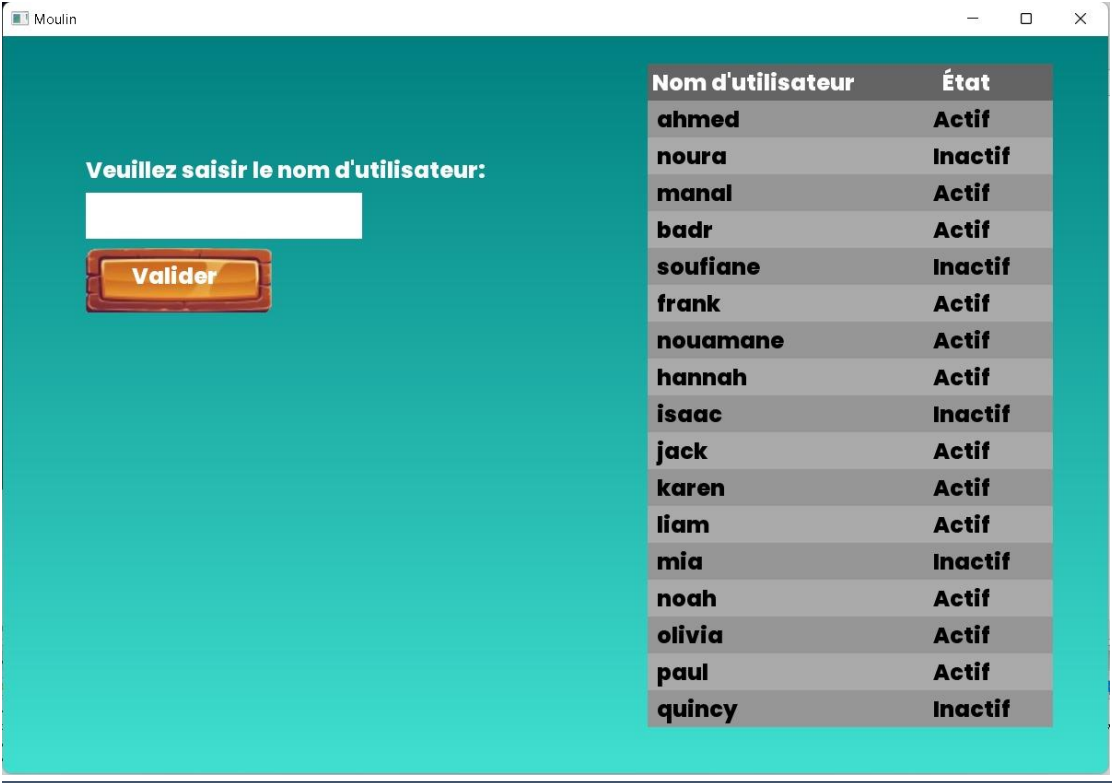
Nom:

Prenom:

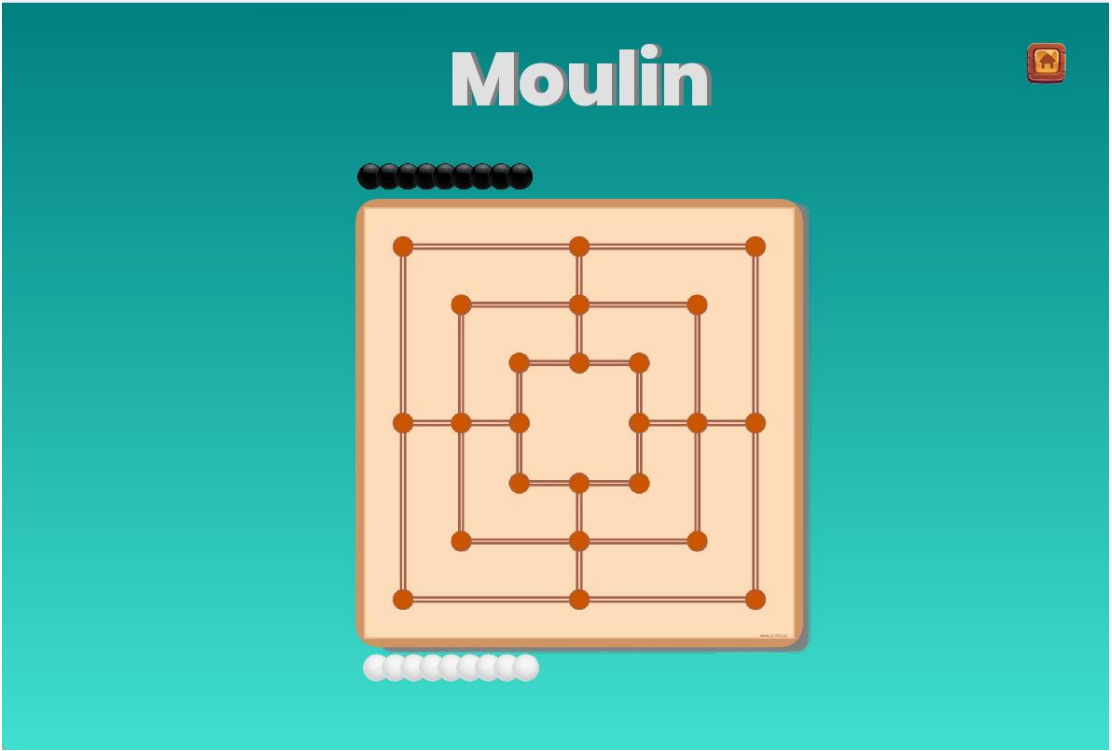
Username:

Mot de passe:

Interface pour choisir l'adversaire en ligne:



Interface du jeu:

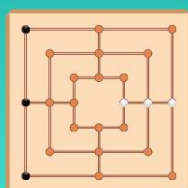


Interface des règles du jeu:



1. Le jeu commence avec un plateau vide.
2. Les joueurs doivent placer à tour de rôle leurs pions (neuf pions au maximum) sur un point d'intersection libre du plateau, suivant la couleur qui leur est attribuée.
3. Le but est de faire un 'moulin : trois pions alignés'.
Si un joueur forme un moulin, il retire un pion à son adversaire (en dehors d'un moulin éventuel).
4. Quand les pions sont tous posés, les joueurs peuvent les glisser d'un point d'intersection à un autre point de la ligne (un mouvement à la fois).
5. Quand un joueur n'a que trois pions, il peut sauter d'un point à un autre.
6. Les pions formant un moulin sont intouchables par l'adversaire. Mais s'ils sont déplacés par le propriétaire, ils deviennent vulnérables.
7. Le jeu continue ainsi jusqu'à ce qu'un joueur n'a que deux pions en sa possession. Dans ce cas, son adversaire est déclaré gagnant.

Exemple de deux moulins un avec des pions noirs et l'autre avec des pions blanc :



Conclusion

Le projet de création du jeu de Moulin, enrichi par l'intégration des algorithmes Minimax et Alpha-Beta, et complété par une base de données pour le support des jeux en ligne, a atteint un nouveau niveau de complexité et de fonctionnalité. En développant une IA optimisée capable de jouer stratégiquement, nous avons rendu le jeu compétitif et engageant. L'application console et l'interface graphique, réalisées avec SFML, offrent des expériences utilisateur variées. La base de données pour les jeux en ligne permet aux utilisateurs de jouer et de sauvegarder leurs parties, facilitant ainsi une expérience multijoueur enrichie. Ce projet, en surmontant les défis techniques liés à l'encodage et à l'optimisation, pose les bases d'un jeu de Moulin robuste, évolutif et socialement interactif, ouvrant des perspectives excitantes pour des fonctionnalités futures .